NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

DARPA

Self-Adaptive Discovery Mechanisms for Improved Performance in Fault-Tolerant Networks

Kevin Mills, Doug Montgomery, Scott Rose,
Stephen Quirolgico, Kevin Bowers, and Chris Dabrowski

DARPA FTN PI Meeting
July 25, 2002

Scalable Software for Military Environments

Presentation Outline

- One-Page Review of Project Objective and Plan
- One-Page Refresher on Service Discovery Protocols
- Scalability Questions about Universal Plug-and-Play (UPnP) M-Search (*investigated through simulation results*)
- Adaptive Jitter Control for UPnP M-Search
  - Four Algorithms: Random Burst (RB), Random Paced (RP), Scheduled Burst (SB), Scheduled Paced (SP)
  - Performance characteristics (*obtained via simulation results*)
- Summary of Other Accomplishments Since January 2002
- Plan for Next Six Months
- Conclusions

Project Objective

Research, design, evaluate, and implement self-adaptive mechanisms to improve performance of service discovery protocols for use in fault-tolerant networks.

Project Plan – Three Phases

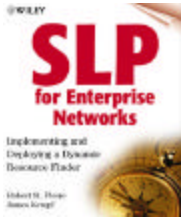
- Phase I – characterize performance of selected service discovery protocols (Universal Plug-and-Play – UPnP – and Jini) as specified and implemented
  - develop simulation models for each protocol
  - establish performance benchmarks based on default or recommended parameter values and on required or most likely implementation of behaviors
- Phase II – design, simulate, and evaluate self-adaptive algorithms to improve performance of discovery protocols regarding selected mechanisms
  - devise algorithms to adjust control parameters and behavior in each protocol
  - simulate performance of each algorithm against benchmark performance
  - select most promising algorithms for further development
- Phase III – implement and validate the most promising algorithms in publicly available reference software

Dynamic Discovery Protocols in Essence

Dynamic discovery protocols enable *network elements*:

(1) to *discover* each other without prior arrangement,
(2) to *express* opportunities for collaboration,
(3) to *compose* themselves into larger collections that cooperate to meet an application need, and
(4) to *detect and adapt to changes* in network topology.

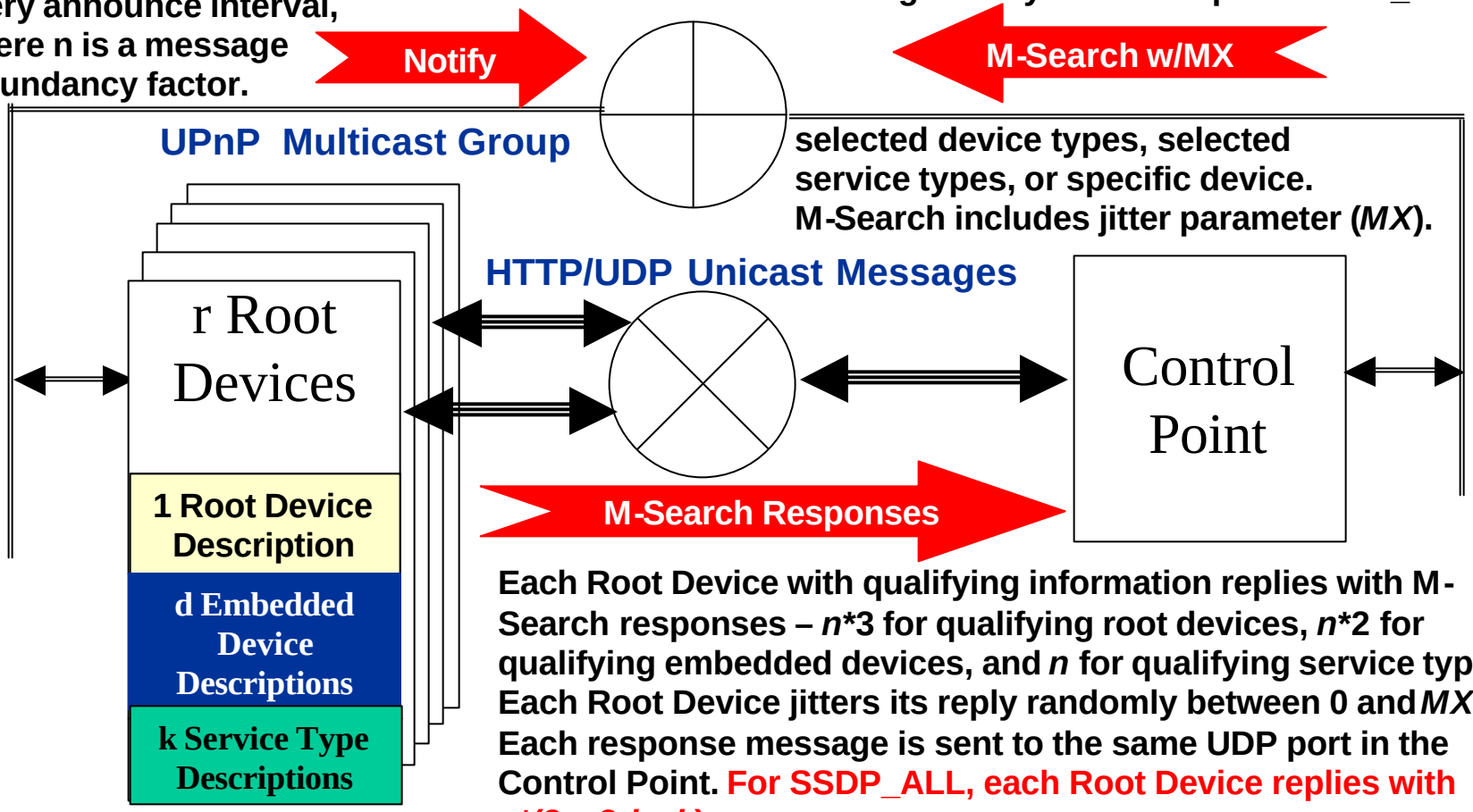
Selected First-Generation Dynamic Discovery Protocols

JINI 3-Party Design	UPnP FORUM 2-Party Design	 Adaptive 2/3-Party Design
The Salutation Consortium Vertically Integrated 3-Party Design	HAVi Network-Dependent 3-Party Design	Bluetooth™ Network-Dependent 2-Party Design

UPnP: ATwo-Party Architecture with Long Service-Announcement Intervals (≥ 30 mins) and Compensating M-Search Queries

Each Root Device multicasts *n* * (3 + 2*d* + *k*) Notify msgs. every announce interval, where n is a message redundancy factor.

Announce intervals are ≥ 30 minutes, so Control Point may multicast M-Search message at any time to request SSDP_ALL,



Each Root Device with qualifying information replies with M-Search responses – *n3 for qualifying root devices, *n**2 for qualifying embedded devices, and *n* for qualifying service types. Each Root Device jitters its reply randomly between 0 and *MX* s. Each response message is sent to the same UDP port in the Control Point. For SSDP_ALL, each Root Device replies with *n**(3 + 2*d* + *k*) responses.**

Scalability Questions about UPnP M-Search

- How does UPnP M-Search perform as network scale varies in terms of Root Devices (*r*), Embedded Devices per Root Device (*d*), unique Service Types per Root Device (*k*), and message redundancy factor (*n*)?
- Can self-adaptive mechanisms improve M-Search performance in response to changing network scale?
- Can self-adaptive mechanisms achieve optimal M-Search performance?

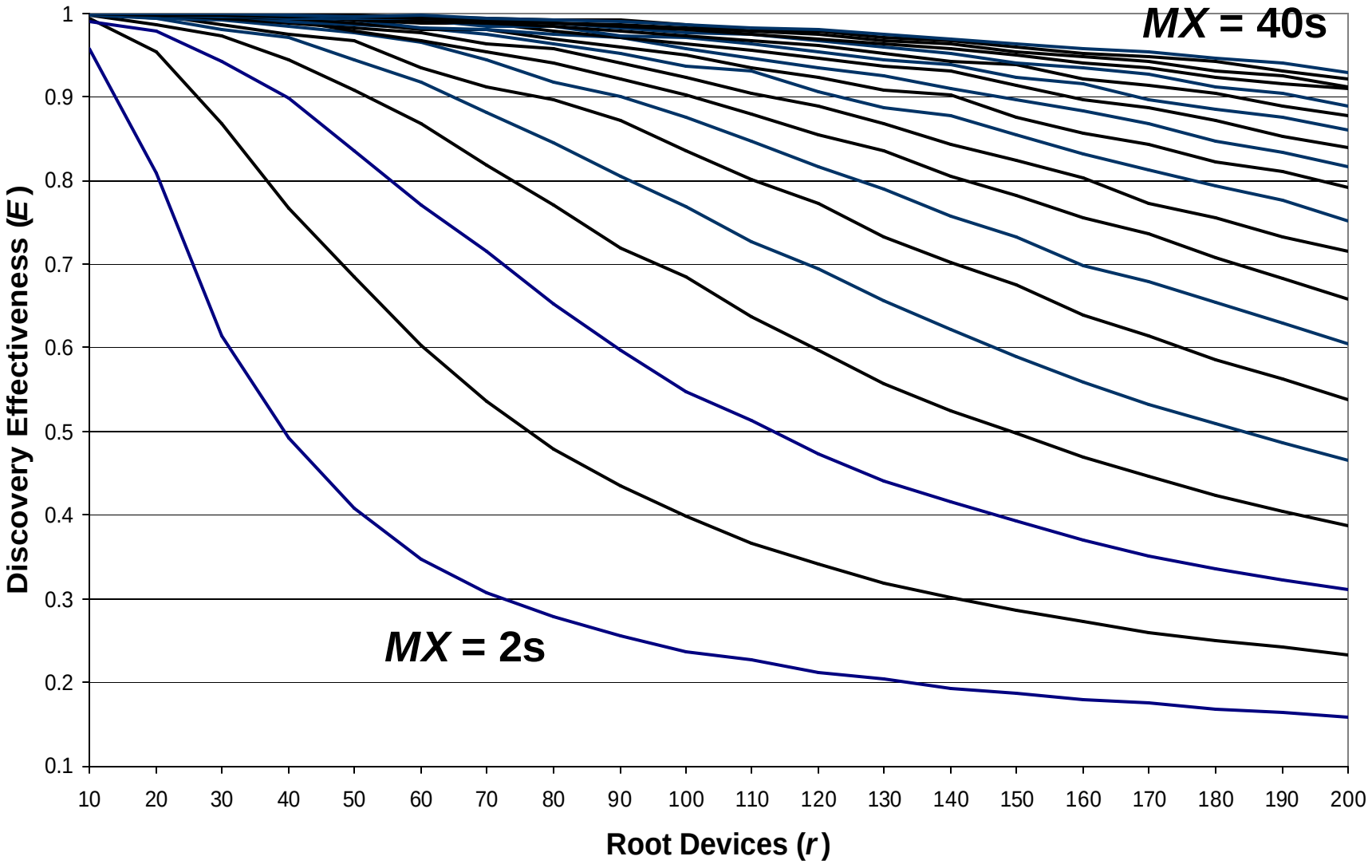
How does UPnP M-Search Perform as Network Scale Varies?

- A Control Point joins an already operating UPnP network and desires to gain a full picture of the topology; thus, issues an M-Search for SSDP_ALL.
- Assume that the Control Point M-Search task runs every 5 ms and can process only one message during each time slice. This means that the Control Point M-Search task can process around 200 messages per second.
- Assume that the M-Search task can occupy at most 40 message buffers.
- Let the number of root devices (*r*) vary from 10 to 200 in increments of 10, and let each root device contain 2 embedded devices (*d* = 2) and 3 unique service types (*k* = 3). Let the message redundancy factor be 2 (*n* = 2).
- The Control Point must choose an *MX* value to include in the M-Search. Since the best *MX* value is not known, let *MX* vary from 2 s to 40 s in 2-s increments.
- For each combination of network size (*r* * *d* * *k*) and *MX* value, measure M-Search performance as experienced at the Control Point.

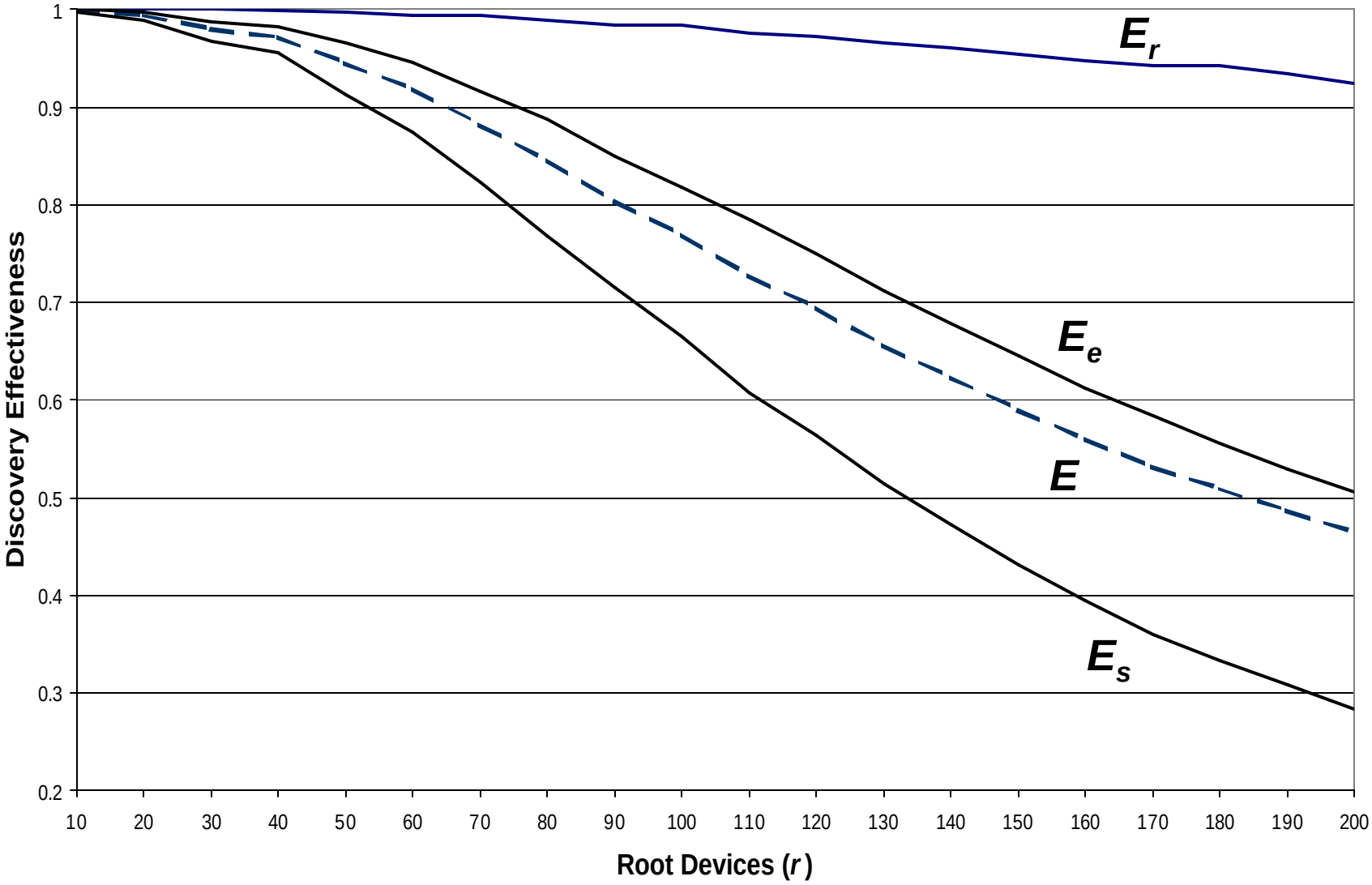
Performance Metrics for UPnP M-Search

- Discovery Effectiveness (*E*): probability that a particular entity, independent of its class, is discovered by the Control Point. Also, discovery effectiveness for specific entity classes: E_r – Root Devices, E_e – Embedded Devices, and E_s – Services.
- Discovery Latency (*L*): – time between successive discovery of new entities by the Control Point.
- Buffer Occupancy (*B*): proportion of available buffers containing messages awaiting processing by the Control Point M-Search task.
- Control Point CPU Usage (CP_{cpu}): CPU seconds per message used by the Control Point to process incoming M-Search responses.

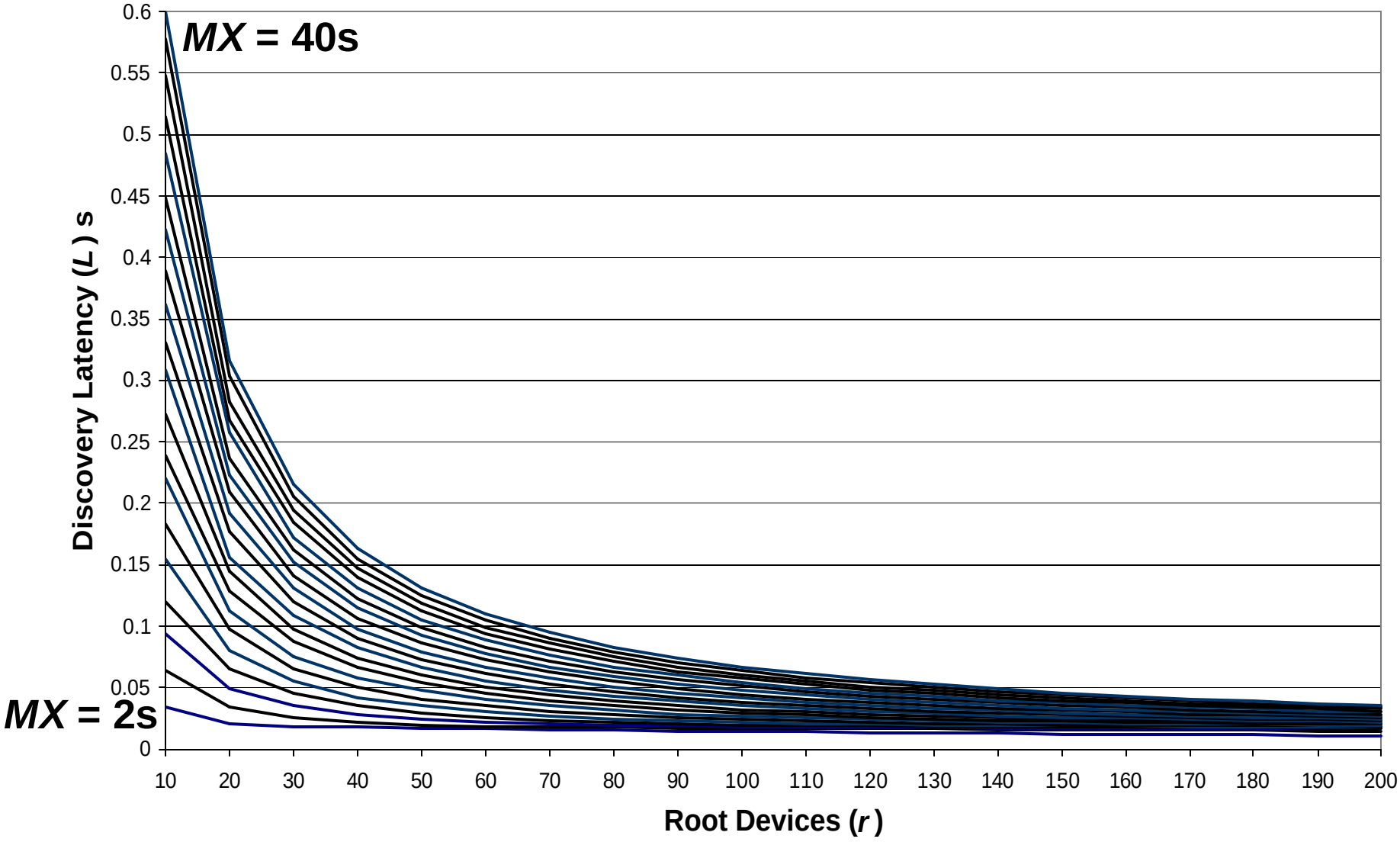
Discovery Effectiveness vs. Network Size for Various MX Values



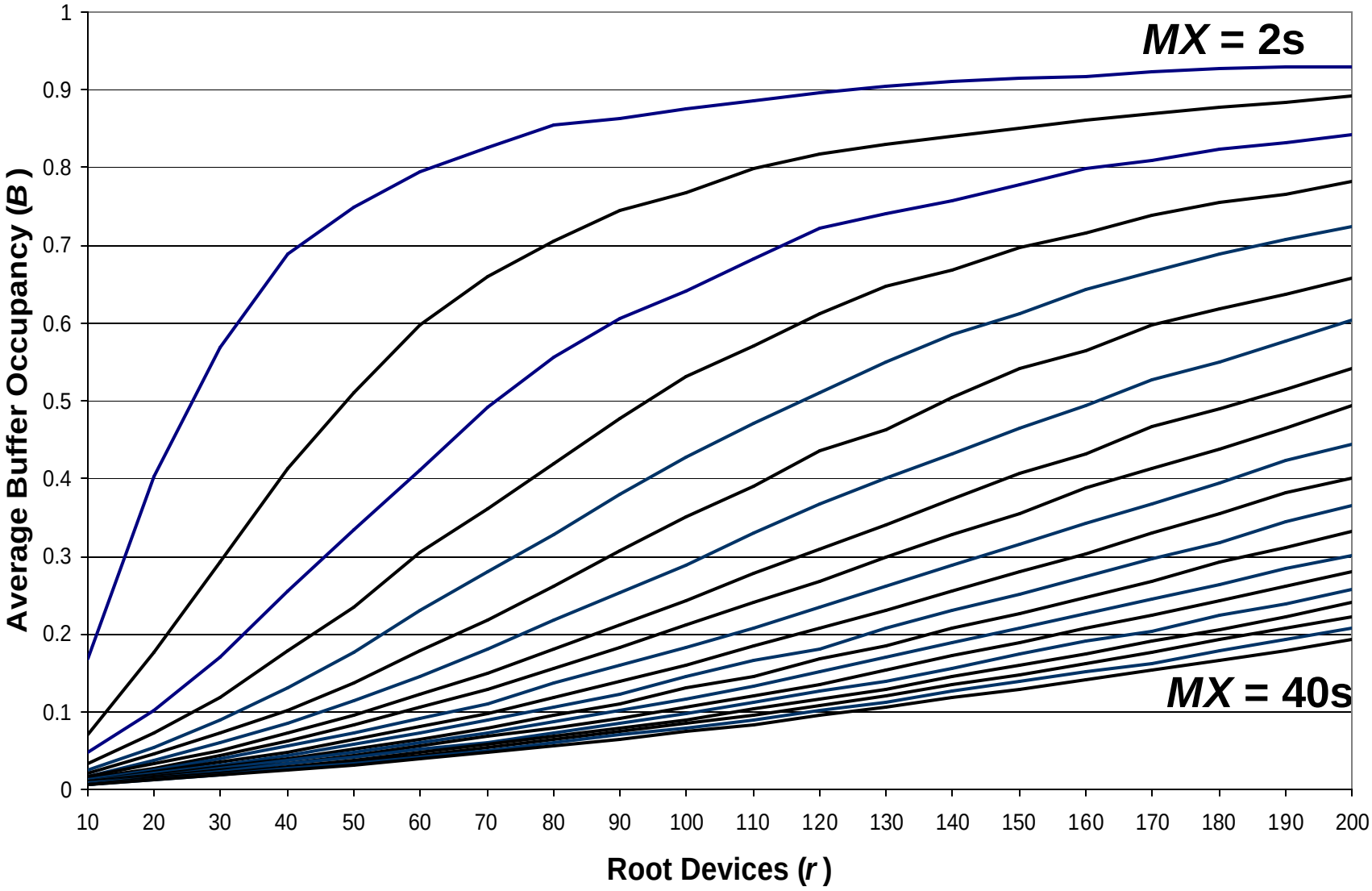
Discovery Fairness vs. Network Size (MX = 10 s)



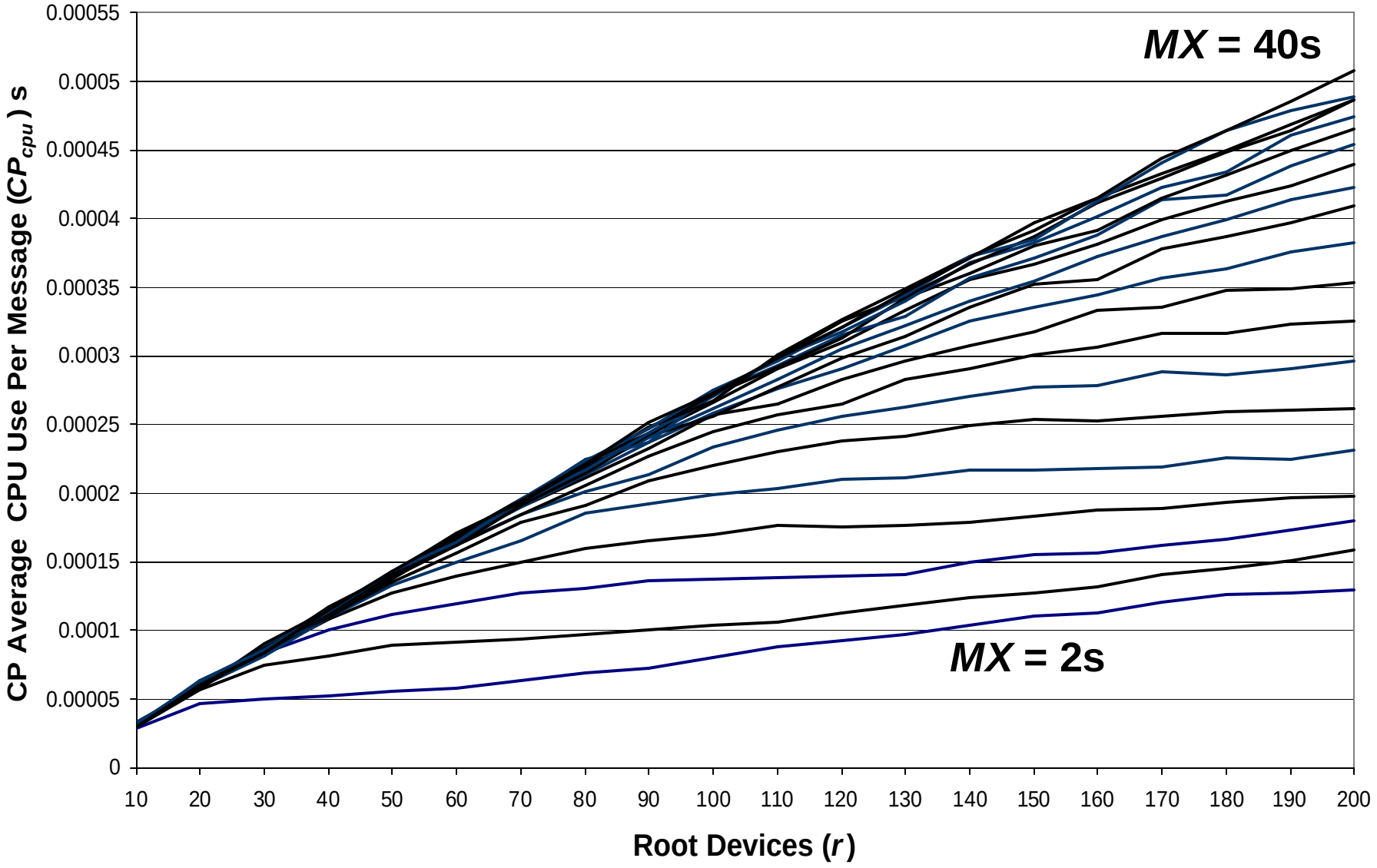
Discovery Latency vs. Network Size for Various MX Values



Buffer Occupancy vs. Network Size for Various MX Values



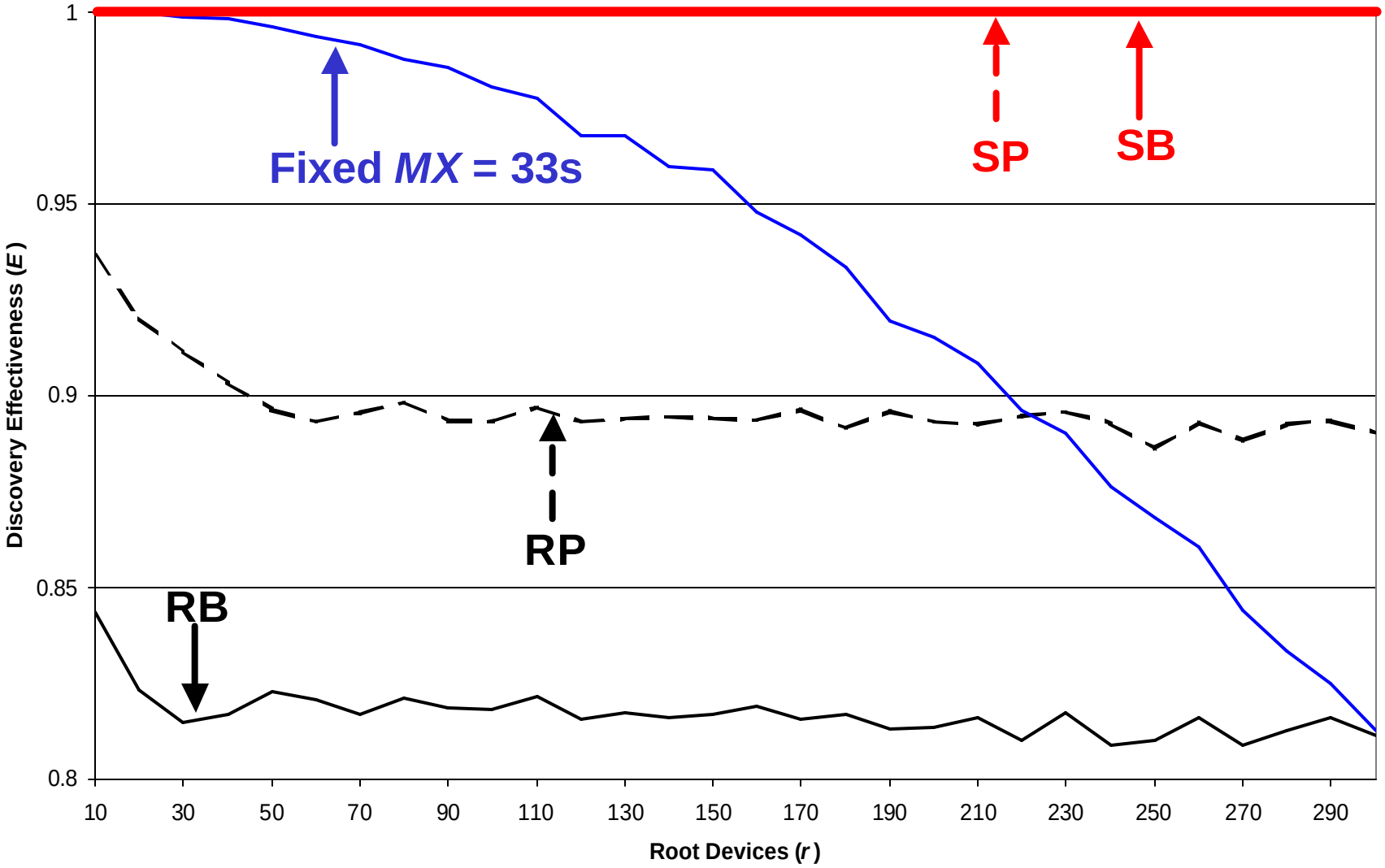
CP CPU Use vs. Network Size for Various MX Values



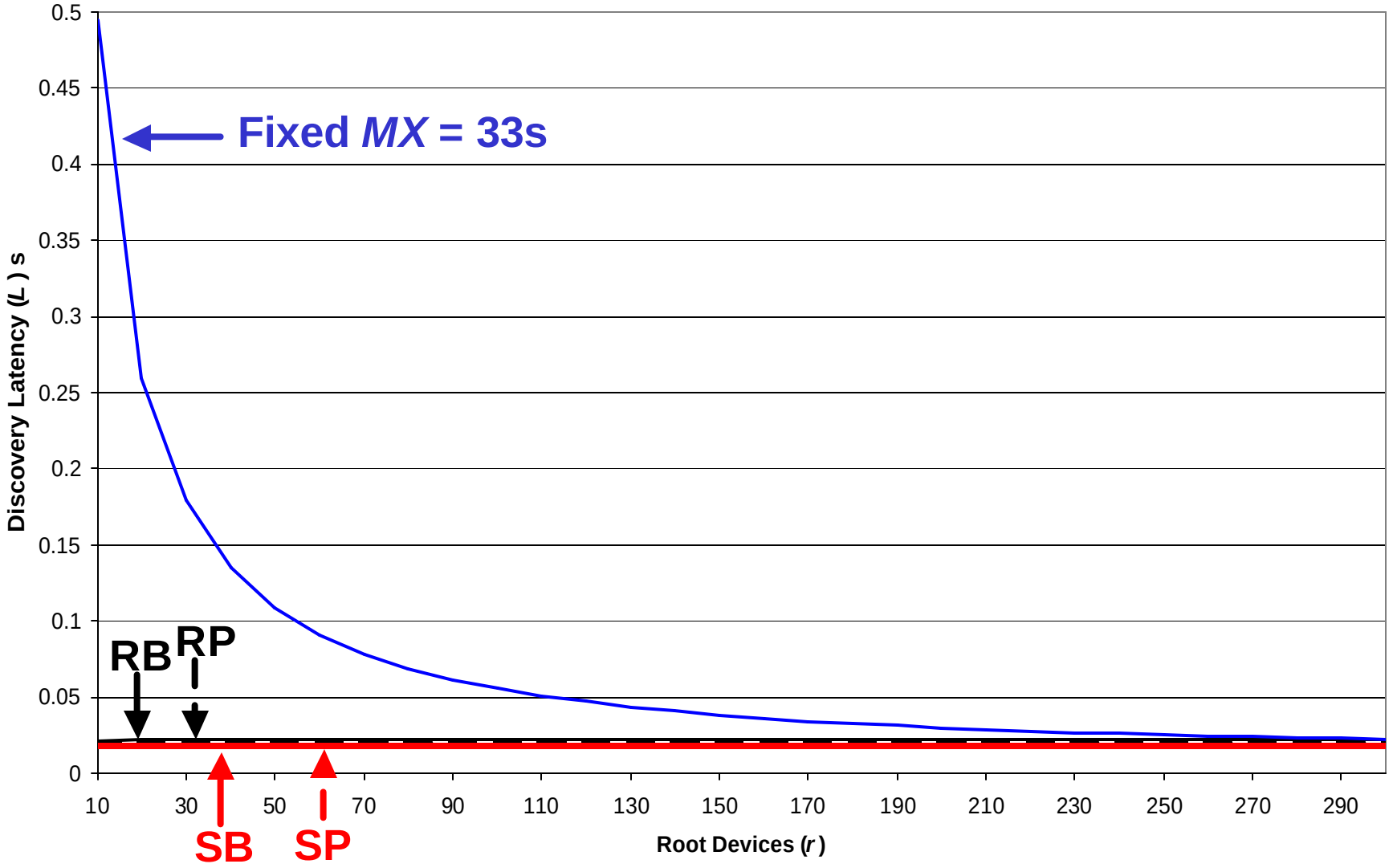
Can Self-Adaptive Mechanisms Improve M-Search Performance?

- **Suppose Root Devices listen to announcements and build a network map (NM).**
- **Suppose Control Points issue M-Searches containing a rate parameter (*R*) that represents the number of msgs/sec that the M-Searcher can process.**
- **Given NM and *R*, each Root Device can compute a maximum jitter value (*Jstart*) and a time (*Jend*) when the last response should arrive at the Control Point.**
- **Each Root Device can select a random time uniformly distributed between 0 and *Jstart* to send its M-Search response messages.**
- **Alternatively, assuming Root Devices will send responses in order based on increasing value of their unique identities, given NM and *R*, each Root Device can compute a scheduled time (*Stx*) to transmit its M-Search responses.**
- **Root Devices can send responses in a burst or paced at rate R, leading to four possible adaptive algorithms: RANDOM BURST (RB), RANDOM PACED (RP), SCHEDULED BURST (SB), and SCHEDULED PACED (SP).**
- **In any of the four algorithms, each Root Device includes *Jend* in each response message, so that the Control Point will know how long to listen.**

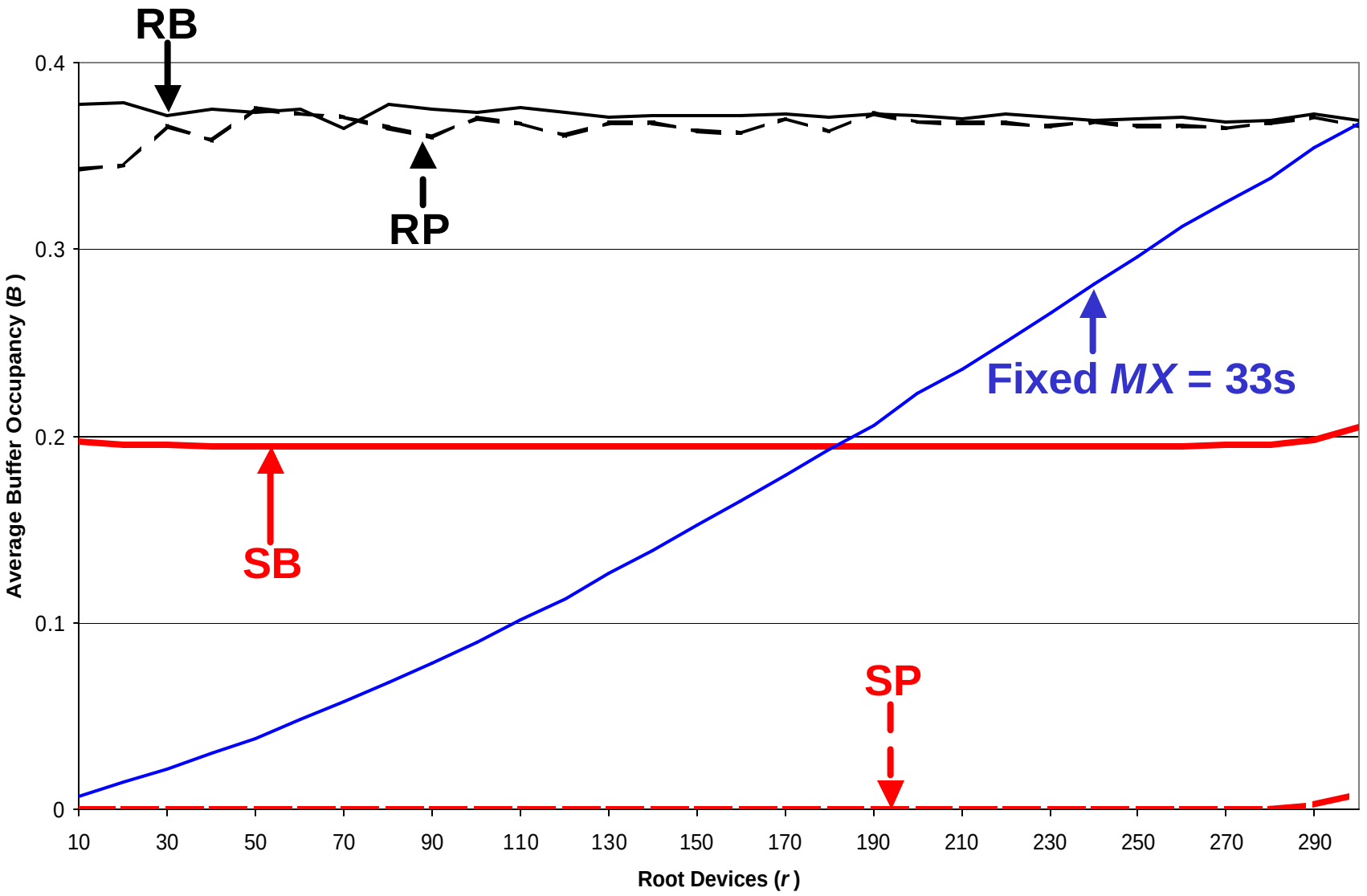
Discovery Effectiveness vs. Network Size for Various Algorithms



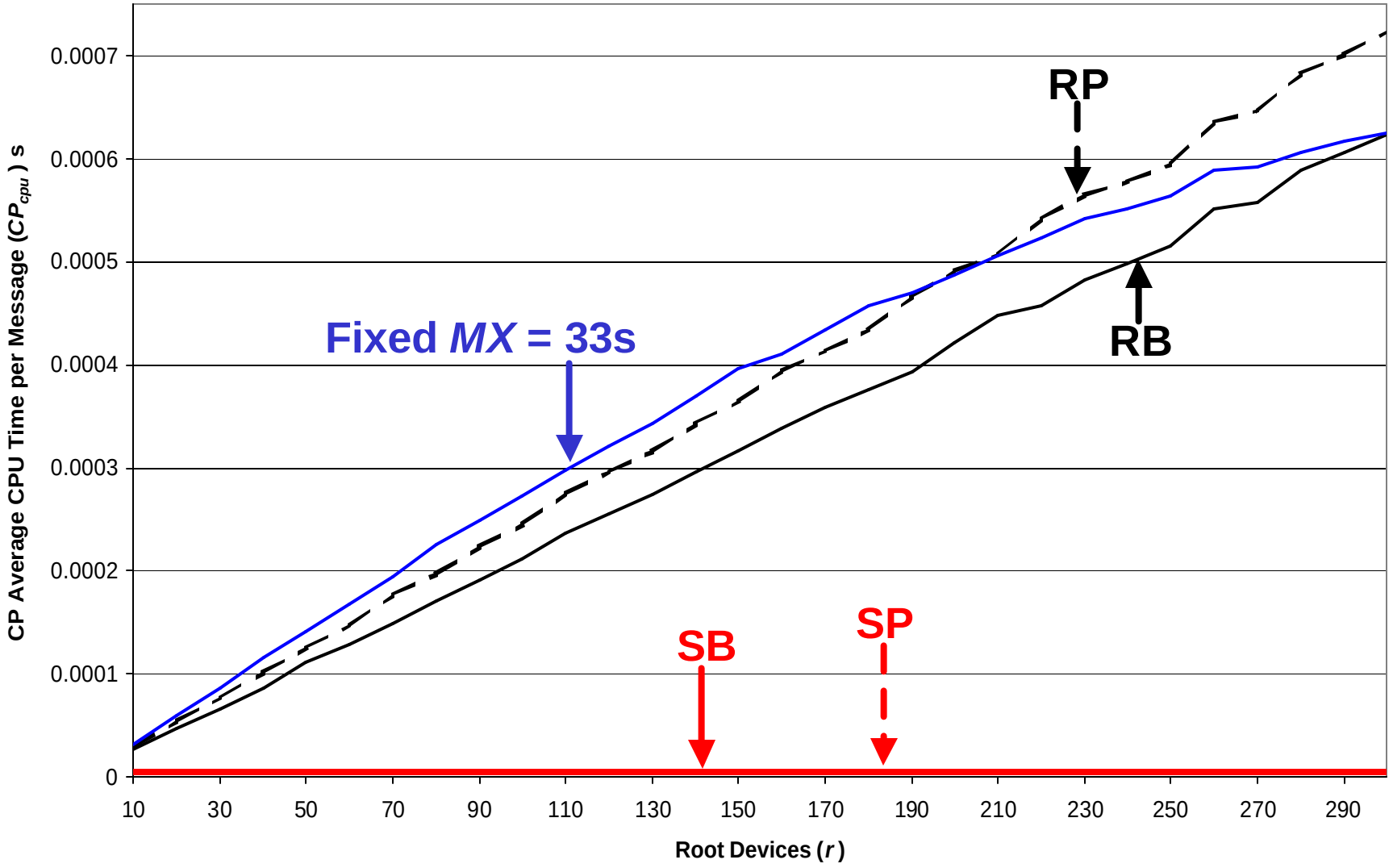
Discovery Latency vs. Network Size for Various Algorithms



Buffer Occupancy vs. Network Size for Various Algorithms



CP CPU Use vs. Network Size for Various Algorithms



Costs, Caveats, and an Alternative Approach

- Adaptive Jitter Control comes with the following costs
  - Each Root Device must listen to announcements and build and store a network map (size~40+**SUM over *r* [**60 + (d_r*12) + (t_r*12) + (k_r *12)**]** =1.2 Kb to 37 Kb in experiments)
  - Each Root Device must compute jitter values for each M-Search received (assuming SSDP_ALL: CPU time~**SUM over *r* [**I_{cpu} * (1 + d_r + k_r)**]** = 0.3 to 9 ms in experiments)

- Adaptive Jitter Control works with the following caveats
  - Using the scheduled approach, collisions will occur unless all Root Devices have perfect knowledge of the network map
  - Perfect knowledge can be assumed to exist during steady-state operation, but Root Devices starting up in an existing network must acquire the network map; for one approach see below (which might also serve as an alternative to M-Search SSDP_ALL)

- An Alternative Approach might be
  - Allow selected Root Devices to offer a network mapping service
  - Control Point uses M-Search to discover existence of network mapping services, and to learn how much information each service knows (this can be included in M-Search responses)
  - If mapping services are found, Control Point uses reliable transport service to request the network map from a mapping service of choice; otherwise, Control Point uses adaptive M-Search
  - Root Devices could use the same approach to discover an initial network map when they start up, then update their copy of the map as changes occur, using their updated local copy to adapt jitter values for incoming M-Search queries

Other Accomplishments Since January 2002

- Published two papers
  - "Understanding Consistency Maintenance in Service Discovery Architectures during Communication Failure", C. Dabrowski, K. Mills, and J. Elder, *Proceedings of the 3rd International Workshop on Software Performance,* Rome, July 24-26, 2002.
  - "Understanding Consistency Maintenance in Service Discovery Architectures in Response to Message Loss", C. Dabrowski, K. Mills, and J. Elder, *Proceedings of the 4th International Workshop on Active Middleware Services*, Edinburgh, July 25, 2002.
- Completed characterization of UPnP and Jini behavior when propagating information during message loss
- Completed scalable (up to 500 nodes) discrete-event simulation model of UPnP – based on source code from Intel's Linux SDK for UPnP - and of Jini – based on source code from Sun's Jini distribution
- Extended our existing generic structural model of service discovery systems to cover behavior, message vocabulary, and consistency conditions

Plan for the Next Six Months

- Draft two journal papers: (1) characterizing the behavior of Jini and UPnP in hostile environments (jamming, congestion, and cyber attack) and (2) proposing a verifiable generic model for service discovery systems
- Submit two papers on recent results
- Complete characterization of UPnP and Jini behavior *(ending Phase I)*
  - Operation and performance during node failure
  - Performance under increasing network size
- Develop and investigate additional ideas for self-adaptive discovery mechanisms in UPnP and Jini
- Complete scalable discrete-event simulation model of the Service Location Protocol (SLP)

Conclusions

- Emerging industry discovery protocols exhibit performance characteristics that vary based on parameter settings, network size, and resource availability
- Tuning such dynamic systems cannot rely on manual configuration methods
- We illustrated one case – UPnP M-Search – where jitter control values interact with network size to determine performance and resource usage
- We proposed four variations of a self-adaptive jitter control algorithm for UPnP M-Search, and we investigated relative performance among the algorithms
- We explained the costs and assumptions underlying the proposed algorithms
- We suggested an alternative method to learn about the availability of devices and services in a network
- We plan to share our findings with Microsoft and Intel